DEEPMPC: A MIXTURE ABR APPROACH VIA DEEP LEARNING AND MPC

Tianchi Huang[1], *Lifeng Sun*[1,2]

[1]BNRist, Department of Computer Science and Technology, Tsinghua University
[2]Key Laboratory of Pervasive Computing, Ministry of Education

ABSTRACT

The leading adaptive bitrate (ABR) algorithm leverages model predictive control (MPC) method to determine next chunks' video bitrate, while it heavily relies on the accuracy of throughput prediction, which thereby fails to perform well in all considered network scenarios. In this paper, we propose *DeepMPC*, which enhances MPC via two deep learning-based modules, i.e., DL-based Throughput Predictor (DTP), which can precisely predict future bandwidth, and Discounted Factor Optimizer (DFO), which estimates the prediction error. Using trace-driven experiments, we illustrate that DeepMPC outperforms existing ABR schemes in all considered network conditions, with the improvements on average QoE of 5.91% - 56.1%. Moreover, we implement DeepMPC in real-world network environments and extensive experimental results demonstrate the superiority of DeepMPC against existing state-of-the-art approaches.

***Index Terms*—** Adaptive Video Streaming, Model Predictive Control, Deep Learning, Reinforcement Learning.

1. INTRODUCTION

Internet video streaming and downloads will grow to more than 82% of all consumer Internet traffic by 2022 [1]. Adaptive bitrate (ABR) video streaming, the method that dynamically switches download chunk bitrates for restraining rebuffering events as well as obtaining higher video bitrates, has become the popular scheme for providing video streaming services with high quality-of-experience (QoE) to the users [2]. Conventional ABR approaches consider the next chunk's video bitrate via only either current network status [3, 4] or buffer occupancy [5, 6], which leads to obtaining the sub-optimal result. Thus, MPC [7] selects the next chunk's bitrate based on jointly considering current buffer occupancy and throughput prediction, which achieves state-of-the-art schemes among traditional model-based ABR algorithms. However, recent work shows that the MPC is entirely limited by throughput prediction accuracy [7, 8] and the determination of discounted factor [9]. Specifically, MPC utilizes the fixed rules, i.e., the harmonic mean of past throughput measured, and past five chunks' prediction error, to make decisions, which will eventually fail to work well under all network conditions. Thus, state-of-the-art ABR algorithm Pensieve [10] adopts deep reinforcement learning (DRL) to generalize an outstanding ABR policy from scratch. Nevertheless, despite the outstanding improvements that AI-based schemes achieve, such methods are often modeled as a black box, which has a lack of interpretability. Thus, we ask if deep learning (DL) will assist MPC to perform better, and in the meanwhile, MPC will also enhance the interpretability of DL-based methods.

In this paper, we propose DeepMPC, an ABR approach with the fusion of DL and conventional MPC method. DeepMPC is composed of two modules for solving the weakness of existing algorithm: i) *DL-based Throughput Predictor (DTP)*, the DL-based model that predicts future throughput via a sequence of past network status; ii) *Discounted Factor Optimizer (DFO)*, which utilize A2C [11], an efficient deep reinforcement learning (DRL) method, to train a neural network (NN) from scratch for determining the proper discounted factor based on current video player's status and past prediction error. Technically, we first collect a corpus of network datasets including various network conditions for training and validating DeepMPC. Then we leverage a faithful offline ABR simulator to emulate various network environments for training a high-performance DFO via DRL. Finally, we merge these two schemes *DTP+DFO* to a novel ABR algorithm, namely DeepMPC. To that end, unlike end-to-end ABR scheme Pensieve, each module of DeepMPC has a clear sub-goal that can be easily explained.

We evaluate DeepMPC and existing ABR schemes, including learning-based ABR scheme Pensieve [10], model-based ABR approach MPC [7], etc., on both offline ABR simulator and real-world implementation. Trace-driven experimental results illustrate that DeepMPC outperforms the off-the-shelp ABR schemes on all considered network conditions, with the improvements on average QoE of 5.91% - 56.1%. Finally, we also validate DeepMPC in real-world network scenarios. Results indicate that our approach improves the average QoE of 10.56% compared with state-of-the-art learning-based ABR scheme Pensieve. In general, we summarize the contributions as follows:

1. To the best of our knowledge, we are the first to use DL and DRL methods to tap the potential for the MPC-based ABR algorithm.

2. We show that the fusion of DL and MPC is not only more effective and interpretable but also achieves state-of-

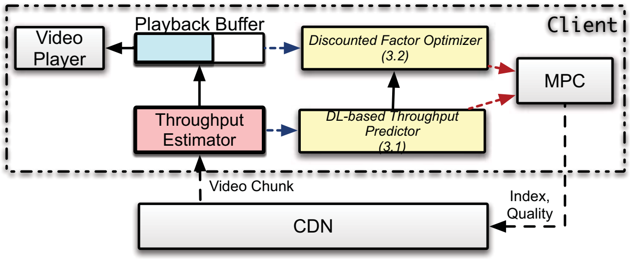


Fig. 1. An Overview of DeepMPC.

the-art performance compared with existing algorithms.

2. DEEPMPC DESIGN

The key idea of MPC is to maximize QoE of users via model predictive control method during the entire session. For each video chunk k, MPC first uses a *throughput predictor* to the estimate future bandwidth C_k. It then calculates the proper video bitrate R_k according to C_k and *current buffer occupancy* B_k, where R_k can obtain the maximum QoE of future *five* chunks. However, traditional MPC methods have their drawbacks, which includes:

1) Inaccurate throughput prediction. Recent work [7] proves that throughput prediction errors have a significant impact on the performance of ABR algorithms. However, prior work only leverages *harmonic mean* of past throughput observed which lacks the precise prediction abilities. Thus, we ask if deep learning can predict further throughput more accurately than previous approaches.

2) Imprecise discounted factor. To counteract the prediction error affected by inaccurate throughput predictor, RobustMPC adopts a discounted factor γ to *underestimate* the throughput for controlling the robustness of the predicted result [7]. In detail, RobustMPC estimate past k prediction error for determining the current factor, which yields a reliable result. However, such heuristic methods require careful tuning, which fails to provide high QoE in all considered network conditions [10]. We, therefore, aim to use deep reinforcement learning (DRL) to *provide* a proper discounted factor γ for any network status.

To this end, motivated by the recent success of DL on estimating future bandwidth tasks [12], we propose DeepMPC, aiming to leverage DL for accurately predicting future throughput, so as to improve the overall QoE performances of MPC. As shown in Figure 1, our approach picks the proper bitrate for the next chunk k with the methods as follows: 1) **DL-based Throughput Predictor** (§2.1), which utilizes DL to predict future throughput C_k from past throughput; 2) **Discounted Factor Optimizer** (§2.2), which uses DRL to determine the discount factor γ_k for chunk k, and calculates the final throughput predicted $\hat{C}_k = \gamma_k * C_k$; 3) **Conventional MPC Model**, which further computes best bitrate R_k for the next chunk.

2.1. DL-based Throughput Predictor

The model will estimate throughput c_{k+1} of video chunk $k+1$ from the given time series of past throughput observed C_k.

Inputs & Outputs. The throughput predictor takes past time t chunks' throughput vector $C_k = \{c_{k-t+1}, \dots, c_k\}$ into NN, where c_i is the normalized throughput for video chunk i. The predictor takes future p chunks' average throughput as the output.

NN Architecture. The input first passes a 1D-CNN layer with 64 filters, each of size 3 with stride 1. Next, it then passes a fully connected layer with 32 filters. Finally, it outputs as a single value of $(0, 1)$,

Loss Function. We adopt mean square error (mse) to evaluate the gap between value predicted y and the ground truth $\hat{y}$.

2.2. Discounted Factor Optimizer

In our study, we propose Discounted Factor Optimizer (DFO), which uses DRL to *determine* a proper discounted factor γ for the given state.

State. For each chunk k, DFO takes $S_k = \{B_k, E_k, D_k\}$ as the input, in which C_k, E_k, D_k are vectors that represent past t chunks' buffer occupancy, throughput measured, throughput prediction error and download time respectively. The throughput prediction error is computed as $Err = \frac{|pred - \hat{real}|}{\hat{real}}$, where $pred$ means the next chunk's download throughput predicted by the throughput predictor and $\hat{real}$ is the next chunk's throughput measured.

Action. In this work, considering the trade-off between NN's convergence time and performance, we pick *10* actions $A = \{0.1, \dots, 1.0\}$ to represent the DFO's action.

Reward. we leverage QoE_{lin} as reward for optimizing NN. Details are illustrated in §3.1.

NN Architecture. The DFO's NN architecture is composed of feature extraction layer, combination layer, and regression layer. The NN first passes the state into the feature extraction layer: for the input as a vector, it uses 1D-CNN with stride=1, kernel=3, channel=128 to extract features; for the input as a value, it leverages fully-connected with 128 neurons to ascension dimensions. Then the output of the extraction layer is combined with the combination layer. Finally, the output is computed by a fully-connected with 128 neurons.

Training Methodology. We use A2C [11], a state of the art actor-critic DRL algorithm, to train DFO.

Implementation. We leverage an AWS in 20 cores to train DFO, and the training time lasts almost 50 hours with 20 agents. We use TensorFlow [13] to implement DFO's NN architecture. Besides, we set the actor network's learning rate $\alpha_a = 0.0001$, critic network's learning rate $\alpha_p = 0.001$, and entropy weight $\beta = 5.0$ down to 0.1 during the training process, as suggested by the authors [9].

3. EVALUATION

3.1. Implementation

▷ **Experimental Testbed Setup.** Our work is composed of two experiments: 1) **Trace-driven offline emulation.** We use Pensieve virtual player, a faithful ABR offline simulator, to evaluate DeepMPC via network traces. The simulator is provided by Mao et al. [14], which is written by Python2.7. 2) **Real-world Deployment.** Meanwhile, we also establish a client-server based full-system implementation. On the server-side, we deploy an HTTP video server. On the client-side, we modify `Dash.js` [15] to implement our video player client. Finally, we implement DeepMPC as a service on the ABR server.

▷ **Video Datasets.** In this paper, we use *EnvivioDash3* from the DASH-246 JavaScript reference client [16], the same video dataset commonly used in [10, 17, 9]. In details, the video is encoded by the H.264 codec at video bitrates in the range of {0.3, 0.75, 1.2, 1.85, 2.85, 4.3} Mbps. The total length of the video is 193 seconds, which is divided into 48 chunks, where each chunk is 4 seconds.

▷ **Network Trace Datasets.** We collect network traces from different public datasets for training and testing DeepMPC. The traces contains HSDPA [18], FCC [19] and Oboe [20], totally 40 hours.

▷ **DTP's Training Set.** We randomly pick the next chunk bitrate from the virtual player and store the information into the dataset, where the dataset contains various network environments. We use 80% dataset for training and 20% for validating.

▷ **QoE Metrics.** In this paper, we use the general QoE metric QoE_{lin} [7, 9, 10], the linear mapping formula which was used by MPC [7], to evaluate existing ABR schemes:

$$\texttt{QoE} = \sum_{n=1}^{N} R_n - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |R_{n+1} - R_n|, \quad (1)$$

where N is the total number of chunks during the session, R_n represents the each chunk's video bitrate, T_n reflects the rebuffering time for each chunk n. We set $\mu = 4.3$ as suggested by [10, 17].

▷ **ABR Baselines.** In this paper, we select several representational ABR algorithms from various type of fundamental principles:

Rate-based [3]: uses harmonic mean of past five throughputs measured as future bandwidth.

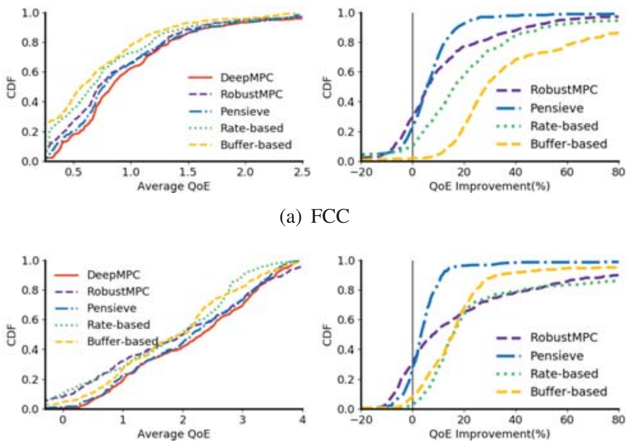
Buffer-based [5]: dynamically picks the next chunk bitrate according to the buffer occupancy.

RobustMPC [7]: inputs the buffer occupancy and throughput predictions, and then maximizes the QoE by solving an optimization problem. In this experiment, we use the *MPC* and *RobustMPC* implementation by ourselves.

Pensieve [10]: utilizes DRL to pick bitrate for next video chunks. We use the pre-trained Pensieve model provided by the authors [14].

Table 1. Performance comparison of throughput prediction models on different network traces. *The lower the better.*

	FCC	HSDPA	Oboe	Model Size(MB)
Harmonic(Baseline)	0.201	0.259	0.159	-
GRU	0.173	0.223	0.140	0.4
Fully-Connected	0.223	0.230	0.214	0.041
Hybrid	**0.172**	0.220	**0.139**	0.32
DTP(1D-CNN)	0.173	**0.218**	0.142	**0.1**



(a) FCC

(b) Oboe

Fig. 2. Comparing DeepMPC with existing ABRs under various network conditions. Results are illustrated with CDF distributions and QoE improvement curves.

3.2. DTP with other NN architectures

In this experiment, we aim to figure out the best network architecture from DTP to the following architectures which are listed as follows: **Harmonic mean.** The default throughput predictor used by conventional MPC [7]; **Gated Recurrent Unit (GRU) [21]:** uses double-layered GRU layers, whose the number of hidden units is 64; **Fully-Connected:** a fully-connected layer with 64 neurons; **DTP:** The NN used in this work (§2.1); The fusion of DTP and fully-connected as well as GRU approach.

We test the performance for each architecture via the trace-driven simulator under different network data traces including FCC, HSDPA and Oboe datasets. Results are summarized as symmetric mean absolute percentage error (sMAPE), which is computed as SMAPE $= \frac{1}{n}\sum_{t=1}^{n} \frac{2|F_t - A_t|}{(|A_t|+|F_t|)}$, where A_t means the ground truth and F_t is the throughput predicted.

As demonstrated in Table 1 we find that DTP outperforms the original harmonic mean method, with the improvements on average accuracy of 13.93%, 15.83%, and 10.69% respectively. Considering the model size for each NN architecture, we find that DTP achieves almost similar performances by using only almost 25% model size of the hybrid scheme.

Table 2. Comparing QoE performance of DeepMPC with other MPC schemes. Results are collected under the *HSDPA*, *FCC* and *Oboe* dataset respectively.

(a) HSDPA

	Original	Robust	DFO
Harmonic	0.30 (67.1%↓)	0.92 (0.5%↓)	0.95 (2.6%↑)
DTP	0.40 (56.8%↓)	0.93 (1.3%↑)	**0.96 (4.0%↑)**
Pensieve	0.92 (-)		

(b) FCC

	Original	Robust	DFO
Harmonic	0.50 (45.7%↓)	0.88 (5.3%↓)	0.97 (4.3%↑)
DTP	0.53 (42.1%↓)	0.95 (2.6%↑)	**0.99 (5.9%↑)**
Pensieve	0.93 (-)		

(c) Oboe

	Original	Robust	DFO
Harmonic	1.95 (6.4%↓)	2.11 (0.7%↑)	2.17 (3.9%↑)
DTP	2.14 (2.6%↑)	**2.20 (5.4%↑)**	2.19 (4.1%↑)
Pensieve	2.09 (-)		

3.3. DeepMPC vs. Existing ABR Schemes

In this part, we attempt to compare the DeepMPC's performance with the recent ABR schemes under several network traces, i.e., FCC, and Oboe. The details of selected ABR baselines are described in §3.1. Figure 2 shows the CDF of QoE metrics on existing methods. We observe that DeepMPC outperforms existing ABR approaches in all considered network scenarios, with the increasing on average QoE of 5.9% to 56.1%. Results also demonstrate that DeepMPC surpasses the state-of-the-art ABR scheme Pensieve, with the improvements on average QoE of 4% to 5.91%. Besides, we also illustrate the CDF of the improvement on QoE of ABRs over DeepMPC. As expected, results illustrate that DeepMPC improves the performance for almost 75% of sessions compared with Pensieve under Oboe dataset. Also, as shown in Figure 2(b), comparing the performance of DeepMPC with Pensieve on the FCC dataset, we find that DeepMPC betters 20% of sessions.

3.4. DeepMPC Ablation Study

In this experiment, we try to investigate how do the proposed modules affect the performance of MPC. The type MPC methods can be concluded as **A +B**, in which model A is for predicting throughput, and model B is to output a discounted factor that aims to recommend a conservative lower bound for future throughput. Specifically, *Harmonic mean+Robust* stands for **RobustMPC** and **DeepMPC** is represented as *DTP+DFO*. The results of the schemes are demonstrated in Figure 3.3. We observe that the harmonic+original MPC scheme works well on Oboe dataset but heavily lacks the performances on HSDPA and FCC dataset. In particular, the DTP+Original MPC scheme performs better than the Pensieve, with the average QoE improving by 2.58%. Comparing with the RobustMPC, the DTP+Robust MPC scheme improves the average QoE by 1.84%-4.6%. Note that it has also already outperformed Pensieve, with the improvements on average QoE of 1.3%-5.4%. Meanwhile, DFO can significantly improve the performance of MPC. Especially, DeepMPC works better than Pensieve on all test network conditions, which improves the average QoE of 4% to 5.91%.

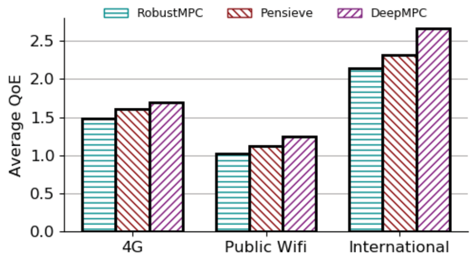


Fig. 3. Comparing the performance of DeepMPC with Pensieve and RobustMPC under various real-world network conditions. Results are shown with average QoE metrics.

3.5. Real-world Experiments

We also set up a real-world experiment to investigate how DeepMPC performs in the wild. In detail, we evaluate the performance of DeepMPC, RobustMPC, and Pensieve under various network conditions including 4G/LTE network, WiFi network and international link (from Singapore to Beijing). For each round, we randomly pick a scheme from ABR scheme candidates and summarize the bitrate selected, rebuffering time and QoE for each chunk. The experiment takes about 2 hours. Figure 3 shows the average QoE results for each scheme under different network conditions. Unsurprisingly, DeepMPC also outperforms previous state-of-the-art ABR scheme Pensieve on average QoE of 5.59%-15.09%.

4. CONCLUSION

In this work, we find that DeepMPC, an ABR scheme that leverages DL and DRL to assist traditional MPC approach, can achieve higher performances compared with previously proposed methods. Experimental results show that the fusion of DL and MPC is successful, which increases the average QoE by 5.91% - 56.1% compared with existing schemes.

Acknowledgement. Let's combat the COVID-19 crisis together! This work is supported by the National Natural Science Foundation of China under Grant 61936011 and 61521002, as well as the Beijing Key Lab of Networked Multimedia (Z161100005016051).

5. REFERENCES

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2016-2021," 2017.

[2] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys & Tutorials*, 2018.

[3] Junchen Jiang, Vyas Sekar, and Hui Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *TON*, vol. 22, no. 1, pp. 326–340, 2014.

[4] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[5] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[6] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM 2016, IEEE*. IEEE, 2016, pp. 1–9.

[7] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *ACM SIGCOMM Computer Communication Review*. ACM, 2015, pp. 325–338.

[8] Yi Sun and et al., "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *SIGCOMM 2016*. ACM, 2016, pp. 272–285.

[9] Zahaib Akhtar and et al., "Oboe: auto-tuning video abr algorithms to network conditions," in *SIGCOMM 2018*. ACM, 2018, pp. 44–58.

[10] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the 2017 ACM SIGCOMM Conference*. ACM, 2017, pp. 197–210.

[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[12] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun, "Delay-constrained rate control for real-time video streaming with bounded neural network," in *NOSSDAV 2018, Amsterdam, Netherlands, June 12-15, 2018*, 2018, pp. 13–18.

[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: A system for large-scale machine learning.," in *OSDI*, 2016, vol. 16, pp. 265–283.

[14] Mao, "hongzimao/pensieve," Jul 2017.

[15] "Dash industry forum — catalyzing the adoption of mpeg-dash," 2019.

[16] "reference client $2.4.0_{2016}$," 2016.

[17] Pablo Gil Pereira, Andreas Schmidt, and Thorsten Herfet, "Cross-layer effects on training neural algorithms for video streaming," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2018, pp. 43–48.

[18] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013, pp. 114–118.

[19] Measuring Fixed Broadband Report, "Raw data measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016, [Online; accessed 19-July-2016].

[20] Usc-Nsl, "Usc-nsl/oboe," Oct 2018.

[21] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv: Neural and Evolutionary Computing*, 2014.